LINUX ADVERTENCIA Y LIMITACIÓN

.

Referencia de este artículo [1].

En el mundo de la computación existen actualmente muchos sistemas operativos a través de los cuales se puede hacer uso de los equipos de cómputo, entre los cuales podemos mencionar Windows, Unix, Minix, Linux etc, cabe mencionar que bajo el nombre de Linux en realidad se engloba a una gran familia de sistemas operativos. En este documento describo una situación que me sorprendió al trabajar con un archivo de texto creado con el editor de msdos y posteriormente tratar de utilizar dicho archivo en un sistema GNU/Linux.

1 Primera edición del archivo

Leyendo una revista llamada USERS LINUX, encontré el listado de código fuente bash, mostrado en la figura 1, para la creación de un guión de shell bash.

```
#!/bin/bash
DIR_ORIGEN=/mnt/disco_lejano_y_seguro
DIR_DESTINO=/home/mi_usuario/directorio-
_fundamental
cd $DIR_ORIGEN
for ARCHIVO in *
do
 ARCHIVO_DESTINO="$DIR_DESTINO/$ARCHIVO"
 if [ -f $ARCHIVO ] && [ $ARCHIVO
     -nt $ARCHIVO_DESTINO]; then
 echo Copiando $ARCHIVO...
 cp $ARCHIVO $ARCHIVO_DESTINO
 fi
done
cd -
```

Figura 1: Código fuente para un guión de Shell Bash

Como comentario, este programa se puede utilizar para realizar un “espejeado” de un directorio. Es decir, supongan que están trabajando en un cierto proyecto de programación en el cual por alguna razón es importante disponer de la versión más reciente de un archivo determinado *filename.txt*, que se supone está contenido en el directorio

/mnt/disco_lejano_y_seguro

Entonces, para asegurarse de que se tiene el archivo *filename.txt* más reciente (cronológicamente) en nuestro directorio

/home/mi_usuario/directorio_fundamental

basta con ejecutar (en un sistema GNU/Linux) el guión de la figura 1, y como consecuencia, si nuestro archivo *filename.txt*, el que ya estaba en el directorio

directorio_fundamental

que es más reciente que el del directorio

disco_lejano_y_seguro

después de la ejecución del guión en nuestro directorio se conservará nuestro archivo *filename.txt*, en caso contrario nuestro archivo *filename.txt* será remplazado por el *filename.txt* del directorio

disco_lejano_y_seguro

En fin, disculpen que no explique para qué sirve este guión de shell pero no es ese el motivo principal de este documento. El punto es que para tratar de usar este guión de shell edité el archivo de la figura 1, utilizando para ello el editor edit de microsoft, disponible en una de las tantas computadoras personales de la UPIITA. Guardando el archivo en un disco de 3.5 pulgadas, con el nombre de Espejo.txt.

2 Segunda edición del archivo

Posteriormente copié el archivo Espejo.txt al disco duro de una PC en la que está instalado el sistema operativo Debian GNU/Linux. Ya con el archivo en el sistema Linux, como es natural, traté de usar el guión de Shell inmediatamente. Pero a pesar de que no existía ningún error de sintaxis, el sistema simplemente no era capaz de ejecutar el guión. Después de varias horas, simplemente decidí editar nuevamente el archivo, pero ahora directamente en el sistema Linux, utilizando para ello el editor vim. Después de guardar el archivo con el nombre Espejo.txt, procedí nuevamente a ejecutar el guión y esta vez funcionó perfectamente. ¿Cuál fue la causa de tan diferentes resultados con dos archivos aparentemente iguales? Un bosquejo de la posible respuesta es lo que se intenta describir en la siguiente sección.

3 El problema de la terminación de línea

Algunas semanas después, estudiando el libro [1] encontré una referencia a un problema que puede estar relacionado con la causa de la situación referida en las dos secciones anteriores. Específicamente con los dos archivos de texto Espejo.txt mencionados antes, se presentó una situación que se supone no debería presentarse: dos archivos de texto “idénticos” ejecutados en el mismo sistema producen resultados muy diferentes al tratar de ejecutarlos como guión de Shell. En el libro [1], pag. 33, se puede leer textualmente

“TEX acepta líneas de un archivo de entrada, excluyendo cualquier terminador de línea que podría ser usado. Debido a esto, el comportamiento de TEX en esta etapa no es dependiente del sistema operativo y del terminador de línea que éste usa (CR-LF, LF, o ninguno para almacenamiento de bloques). De la línea de entrada se eliminan los espacios adicionales que se haya intentado colocar. La razón para esto es histórica; tiene que ver con el modo de almacenamiento de bloques de los servidores IBM grandes. Para algunos problemas con los caracteres de fin de línea de computadoras específicas, véase [2]”

La referencia del libro [1] es la [2] de este documento. Tan pronto como me fue posible consulté el artículo [2], y considero pertinente citar lo siguiente

“...Una investigación descubrió el hecho de que en algunas líneas de texto, un carácter de retorno de carro, o <ctrl-M>, ocurrió, en medio, de hecho, inmediatamente antes del texto faltante.”

“Ha sucedido algo que se supone no debería suceder con TEX: el mismo archivo, ejecutado a través de dos implementaciones diferentes de TEX, ha producido diferentes resultados.

Demostramos que este problema es del tipo sistema operativo específico, construyendo el siguiente archivo de prueba y ejecutándolo en tres diferentes implementaciones de TEX disponibles para nosotros:

The quick brownM̂fox jumped over the lazy dog.

Sobre una DEC-20/TOPS-20, usando la implementación Stanford, la salida leyó “The quick brown fox jumped over the lazy dog.” Sobre una VAX/VMS, ambas implementaciones, la Stanford y la Kellerman y Smith, produjeron “The quick brown lazy dog.”

Mientras platicaba con un colega que usa una implementación VAX/Unix Pastel de TEX, le mencioné el problema y el ejecutó el archivo de prueba en su sistema. Esto produjo un tercer resultado: un mensaje de error,

! Text line contains an invalid character.

El libro TEXBook, pag. 343, declara que

“catcode ‘ ˆ ˆ M=5,

lo cual en la página 37 es interpretado como “fin de línea”. Sobre los sistemas con los que estoy más familiarizado (DEC-20/TOPS-20 y VAX/VMS), el marcador de fin de línea tradicional es un par ASCII retorno de carro/line-feed (<*crlf*>) o

M̂Ĵ

En los archivos que he recibido de usuarios de Mactexintosh, el carácter

M̂

parece ser la norma. (Compilar con TEX un archivo Mac inalterado sobre una DEC-20 invariablemente excede el buffer de entrada, así que me he acostumbrado a traducir los

M̂s

de Mac como <*crlf*> cuando recibo un archivo, si es que el autor no ha hecho aun la conversión.)

Es entonces aparente que hay varias interpretaciones del carácter

M̂

y estas parecen ser dependientes del sistema operativo, o al menos dependientes de la implementación.

• La implementación Stanford TOPS-20 termina una línea y supone que el carácter que sigue inicia una nueva; en otras palabras

M̂

por sí mismo, aunque esto no es algo estándar, es equivalente a <*crlf*>

• Las implementaciones Stanford y K& S VAX/VMS dejan de considerar el contenido de la línea (tratando el

M̂

igual que a un %) y busca el siguiente <*crlf*>.

• La implementación Unix Pastel no permite un carácter

M̂,

solo un par <*crlf*>.

• Podría haber otras —esta investigación realmente solo ha comenzado.”

Es muy probable que la respuesta a la pregunta planteada en la sección 2 se encuentre entre los datos mencionados en las citas arriba incluidas. De hecho, el comando od (octal dump: “vaciado total”, véase [3]) permite observar los caracteres no visibles que los editores de texto que se usaron, introducen al final de cada línea en los dos archivos Espejo.txt antes mencionados (el que se creó con el edit de microsoft y el que se creó con el vim de Linux). En efecto, al ejecutar (en GNU/Linux) el comando (véase [3], pag. 44)

od -c Espejo.txt

para el archivo Espejo.txt creado con edit de microsoft se obtiene

0000000 # ! / b i n / b a s h \r \n D I R

0000020 _ O R I G E N = / h o m e / p r

0000040 o f e s o r / 2 0 0 6 / W a n d
0000060 L - 2 0 0 6 0 3 3 0 - 1 0 5 7 \r
0000100 \n D I R _ D E S T I N O = / m e
0000120 d i a / f l o p p y \r \n c d $
0000140 D I R _ O R I G E N \r \n f o r
0000160 A R C H I V O i n * \r \n d o
0000200 \r \n A R C H I V O _ D E S T
0000220 I N O = ”D I R _ D E S T I N O
0000240 / $ A R C H I V O ”\r \n i f
0000260 [- f $ A R C H I V O]
0000300 & & [$ A R C H I V O - n
0000320 t $ A R C H I V O _ D E S T I
0000340 N O] ; t h e n \r \n e c
0000360 h o C o p i a n d o $ A R C
0000400 H I V O \r \n c p $ A R C H
0000420 I V O & A R C H I V O _ D E S
0000440 T I N O \r \n f i \r \n d o n e
0000460 \r \n c d - \r \n
0000470

Mientras que al ejecutar el comando

od -c Espejo.txt

sobre el archivo Espejo.txt creado con el editor vim de Linux se obtiene

0000000 # ! / b i n / b a s h \n D I R _
0000020 O R I G E N = / h o m e / p r o
0000040 f e s o r / 2 0 0 6 / W a n d L
0000060 - 2 0 0 6 0 3 3 0 - 1 0 5 7 \n D
0000100 I R _ D E S T I N O = / m e d i
0000120 a / f l o p p y 0 \n c d $ D I
0000140 R _ O R I G E N \n f o r A R C
0000160 H I V O i n * \n d o \n A
0000200 R C H I V O _ D E S T I N O = ”
0000220 $ D I R _ D E S T I N O / $ A R
0000240 C H I V O ”\n i f [- f
0000260 $ A R C H I V O] & & [
0000300 $ A R C H I V O - n t $ A
0000320 R C H I V O _ D E S T I N O]
0000340 ; t h e n \n e c h o C o

0000360 p i a n d o $ A R C H I V O .

0000400 . . \n c p $ A R C H I V O

0000420 $ A R C H I V O _ D E S T I N

0000440 O \n f i \n d o n e \n c d -

0000460 \n

0000461

Lo cual nos muestra que el editor edit de microsoft introduce una terminación de línea del tipo <*crlf*>, como lo indican la presencia de los caracteres \r \n al final de cada línea del archivo Espejo.txt en el penúltimo listado, a diferencia del carácter \n al final de cada línea del archivo Espejo.txt en el último listado. En mi opinión la presencia del carácter \r en el archivo creado en edit de microsoft es la causa de que este archivo de texto no pueda ejecutarse como guión de shell a pesar de que tenga el atributo de archivo ejecutable en Linux. Hago una atenta invitación a los lectores para que en caso de que lo consideren pertinente, envíen comentarios al respecto de esta situación, es decir, la causa del problema citado en la sección 2. Realmente lo agradecería mucho.

4 Conclusiones

En mi opinión, los datos mencionados en las citas y en este documento serán de interés para la gente que en la actualidad está intentando empezar a utilizar un sistema operativo diferente del ya muy conocido y utilizado Microsoft Windows.

Referencias

[1] Victor Eijkhout. *Tex by Topic*, a TEX*nician's reference*, Addison-Wesley, 1993. ISBN 0-201-56882-9.

[2] Barbara Beeton. *Controlling <ctrl>-M; ruling the depths*. TUGboat, 9:182-183, 1988. 33.

[3] Brian W. Kernighan, Rob Pike, *El entorno de programación Unix*, Prentice Hall, 1987.